A simple reconstruction of GPSG

A Simple Reconstruction of GPSG

Stuart M. Shieber
Artificial Intelligence Center
SRI International
and
Center for the Study of Language and Information
Stanford University

Abstract

Like most linguistic theories, the theory of generalized phrase structure grammar (GPSG) has described language axiomatically, that is, as a set of universal and language-specific constraints on the well-formedness of linguistic elements of some sort. The coverage and detailed analysis of English grammar in the ambitious recent volume by Gazdar, Klein, Pullum, and Sag entitled *Generalized Phrase Structure Grammar* [2] are impressive, in part because of the complexity of the axiomatic system developed by the authors. In this paper, we examine the possibility that simpler descriptions of the same theory can be achieved through a slightly different, albeit still axiomatic, method. Rather than characterize the well-formed trees directly, we progress in two stages by procedurally characterizing the well-formedness axioms themselves, which in turn characterize the trees.

1 Introduction[1]

Like most linguistic theories, the theory of generalized phrase structure grammar (GPSG) has described language axiomatically, that is, as a set of universal and language-specific constraints on the well-formedness of linguistic elements of some sort. In the case of GPSG, these elements are trees whose nodes are themselves structured entities from a domain of *categories* (a type of *feature structure* [6]). The proposed axioms have become quite complex, culminating in the ambitious recent volume by Gazdar, Klein, Pullum, and Sag entitled *Generalized Phrase Structure Grammar* [2]. The coverage and detailed analysis of English grammar in this work are impressive, in part because of the complexity of the axiomatic system developed by the authors.

In this paper, we examine the possibility that simpler descriptions of the same theory can be achieved through a slightly different, albeit still axiomatic, method. Rather than characterize the well-formed trees driectly, we progress in two stages by procedurally characterizing the well-formedness axioms themselves, which in turn characterize the trees. In particular, we give a procedure which converts GPSG grammars into grammars written in a unification-based formalism, the PATR-II formalism developed at SRI International (henceforth PATR) [5], which has its own declarative semantics, and which can therefore be viewed as an axiomatization of string well-formedness constraints.[2]

The characterization of GPSG thus obtained is simpler and better defined than the version described by Gazdar et al. The semantics of the formalism is given directly through the reduction to PATR. Also, the PATR axiomatization has a clear constructive interpretation, unlike that used in Gazdar et al., thus making the system more amenable to computational implementation. Finally, the characteristics of the compilation—the difficulty or ease with which the various devices can be encoded in PATR—can provide a measure of the expressiveness and indispensability of these devices in GPSG.

2 The GPSG Axioms

2.1 A Summary of the Principles

GPSG describes natural languages in terms of various types of constraints on local sets of nodes in trees. Pertinent to the ensuing discussion are the following:

- ID (immediate dominance) rules, which state constraints of *immediate dominance* among categories;
- metarules, which state generalizations concerning classes of ID rules;
- LP (linear precedence) rules, which constrain the linear order of sibling categories;
- feature cooccurrence restrictions (FCR), which constrain the feature structures as to which are permissible categories;
- feature specification defaults (FSD), which provide values for features that are otherwise unspecified;

and, most importantly,

[1]This research was made possible by a gift from the System Development Foundation.

I am indebted to Lauri Karttunen and Ray Perrault for their comments on earlier drafts, and to Roger Evans, Gerald Gazdar, Ivan Sag, Henry Thompson, and members of the Foundations of Grammar project at the Center for the Study of Language and Information for their helpful discussions during the development of this work.

[2]However, a caveat is in order that the detailed analysis from this perspective of the full range of GPSG devices (especially immediate dominance (ID) rules, and feature cooccurrence restrictions) is not discussed fully here, nor do I completely understand them. (See Section 3.4.) And while in a confessional mood, I should add that the algorithm given here has not actually been implemented.

- universal feature instantiation principles, which constrain the allowable local sets of nodes in trees; these feature instantiation principles include the head feature convention (HFC), the foot feature principle (FFP), and the control agreement principle (CAP).

In GPSG all of these constraints are applied simultaneously. A local set of nodes in a tree is admissible under the constraints if and only if there is some base or derived ID rule (which we will call the *licensing rule*) for which the parent node's category is an extension of the left-hand-side category in the rule, and the children are respective extensions of right-hand-side categories in the rule, and, in addition, the set of nodes simultaneously satisfies all of the separate feature instantiation principles, ordering constraints, etc. By *extension*, we mean that the constituent has all the feature values of the corresponding category in the licensing rule, and possibly some additional feature values. The former type of values are called *inherited*, the latter *instantiated*.

The feature instantiation principles are typically of the following form: if a certain feature configuration holds of a local set of nodes, then some other configuration must also be present. For instance, the antecedent of the control agreement principle is stated in terms of the existence of a *controller* and *controllee* which notions are themselves defined in terms of feature configurations. The consequent concerns identity of agreement features.

2.2 Interaction of Principles

Much care is taken in the definitions of the feature instantiation principles (and their ancillary notions such as controller, controllee, free features, privileged features, etc.) to control the complex interaction of the various constraints. For instance, the FFP admits local sets of nodes with *slash* feature values on parent and child where no such values occur in the licensing ID rule, i.e., it allows *instantiation of slash features*. But the CAP's abovementioned definition of control is sensitive to the value of the *slash* feature associated with the various constituents. A simple definition of the CAP would ignore the source of the slash value, whether inherited, instantiated by the FFP, or instantiated in some other manner. However, the appropriate definition of control needed for the CAP must ignore *instantiated* slash features, but not *inherited* ones. Say Gazdar et al.:

> We must modify the definition of control in such a way that it ignores perturbations of semantic type occasioned by the presence of instantiated *FOOT* features. [2, p. 87]

Thus, the CAP is in some sense blind to the work of the FFP. As Gazdar et al. note, this requirement makes stating the CAP a much more complex task.

The increased complexity of the principles resulting from this need for tracking the origins of feature values is evident not only in the CAP, but in the other principles as well. The head feature convention requires identity of the head features of parent and head child. The features *agr* and *slash*—features that can be inherited from an ID rule or instantiated by the CAP or FFP, respectively—are head features and therefore potentially subject to this identity condition. However, great care is taken to remove such instantiated head features from obligatory manipulation by the HFC. This is accomplished by limiting the scope of the HFC to the so-called *free* head features.

> Intuitively, the *free* feature specifications on a category [the ones the HFC is to apply to] is the set of feature specifications which can legitimately appear on extensions of that category: feature specifications which conflict with what is already part of the category, either directly, or in virtue of the FCRs, FFP, or CAP, are not *free* on that category. [2, p. 95]

That is, the FFP and CAP take precedence (intuitively viewed) over the HFC.

Finally, all three principles are seen to take precedence over feature specification defaults in the following quotation.

> In general, a feature is exempt from assuming its default specification if it has been assigned a different value in virtue of some ID rule or some principle of feature instantiation. [2, p. 100]

Gazdar et al. accomplish this by defining a class of *privileged* features and excluding such features from the requirement that they take on their default value. Of course, instantiated head features, slash features, and so forth are all considered privileged. However, a modification of these exemptions is necessary in the case of lexical defaults, i.e., default values instantiated on lexical constituents. We will not discuss here the rather idiosyncratic motivation for this distinction, but merely note that lexical constituent defaults are to be insensitive to changes engendered by the HFC, as revealed in this excerpt:

> However, this simpler formulation is inadequate since it entails that lexical heads will always be exempt from defaults that relate to their HEAD features.... Accordingly, the final clause needs to distinguish lexical categories, which become exempt from a default only if they covary with a sister, and nonlexical categories, which become exempt from a default if they covary (in relevant respects) with *any* other category in the tree. [2, p. 103]

Thus the interaction of these principles is controlled through complex definitions of the various classes of features they are applicable to. These definitions conspire to engender the following implicit precedence ordering on the principles, principles earlier in the ordering being blind to the instantiations from later principles, which are themselves sensitive to (and exempt from applying to) features instantiated by the earlier principles.[3]

$$\text{CAP} \succ^{4} \text{FFP} \succ \text{FSD}_{lex} \succ \text{HFC} \succ \text{FSD}_{nonlex}$$

Of course, all ID rules, both base and derived are subject to all these principles; yet metarule application is not contingent on instantiations of the base ID rules. Conversely, LP constraints are sensitive to the full range of instantiated features. The precedence ordering can thus be extended as follows:

[3] Current efforts by at least certain GPSG practitioners are placing the GPSG type of analysis directly in a PATR-like formalism. This formalism, Pollard's head-driven phrase structure grammar (HPSG) variant of GPSG, uses a run-time algorithm similar to the one described in this paper [4]. Highly suggestive is the fact that the HPSG run-time algorithm also happens to order the principles in substantially the same way.

[4] We use the symbol $\succ$ to denote one principle "taking precedence over" another.

$$\text{META} \succ \text{CAP} \succ \text{FFP} \succ \text{FSD}_{lex} \succ \text{HFC} \succ \text{FSD}_{nonlex} \succ \text{LP}$$

The existence of such an ordering on the priority of axioms is, of course, not a necessary condition for the coherence of such an axiomatic theory. Undoubtedly, this inherent ordering was not apparent to the developers of the theory, and may even be the source of some surprise to them. Yet, the fact that this ordering exists and is strict leads us to a substantial simplification of the system. Instead of applying all the constraints simultaneously, we might do so sequentially, so that the precedence ordering—the blindness of earlier principles in the ordering to the effects of later ones—emerges simply because the later principles have not yet applied.

This solution harkens back to earlier versions of GPSG in which the semantics of the formalism was given in terms of compilation of the various principles and constraints into pure context-free rules. This compilation process can be combinatorially explosive, yielding vast numbers of context-free rules. Indeed, the whole point of the GPSG decomposition is to succinctly express generalizations about the possible phrasal combinations of natural languages. However, by carefully choosing a system for stating constraints on local sets of nodes—a formalism more compact in its representation than context-free grammars—we can compile out the various principles and constraints without risking this explosion in practice.

The GPSG principles are stated in terms of identities of features. What we need to avoid the combinatorial problems of pure CF rules is a formalism in which such equalities can be stated directly, without generating all the ground instances that satisfy the equalities. What is needed, in fact, is a unification-based grammar formalism [6]. We will use a variant of PATR [5] as the formalism into which GPSG grammars are compiled. In particular, we assume a version of PATR that has been extended by the familiar decomposition into an immediate-dominance and linear-precedence component. This will allow us to ignore the LP portion of GPSG for the nonce.

PATR is ideal for two reasons. First, it is the simplest of the unification-based grammar formalisms, possessing only the apparatus that is needed for this exercise. Second, a semantics for the formalism has been provided, so that, by displaying this compilation, we implicitly provide a semantics for GPSG grammars as well. In the remainder of the paper, we will assume the reader's familiarity with the rudiments of the PATR formalism.

3 The Compilation Algorithm

We postpone for the time being discussion of the metarules, LP constraints, and feature cooccurrence restrictions, concentrating instead on the central principles of GPSG, those relating to feature instantiation. The following nondeterministic algorithm generates well-formed PATR rules from GPSG ID rules. A GPSG grammar is compiled into the set of PATR rules generated by this algorithm.

3.1 Preliminaries

We first observe that a GPSG ID rule is only notationally distinct from an *unordered* PATR rule. Thus, the first step in the algorithm is trivial. For example, the ID rule

$$S \rightarrow X^2, H[-subj] \qquad (R_1)$$

is written in unordered PATR as

$$\begin{array}{l} X_0 \rightarrow X_1,\ X_2 \\ \quad \langle X_0\ n \rangle\ = - \\ \quad \langle X_0\ v \rangle\ = + \\ \quad \langle X_0\ bar \rangle\ = 2 \\ \quad \langle X_0\ subj \rangle\ = + \\ \quad \langle X_1\ bar \rangle\ = 2 \\ \quad \langle X_2\ subj \rangle\ = - \end{array} \qquad (R_2)$$

Note that abbreviations (like S for $[-n, +v, bar2, +subj]$) have been made explicit.

In fact, we will make one change in the structure of categories (to simplify our restatement of the HFC) by placing all head features under the single feature *head* in the corresponding PATR rule. We do not, however, add an analogous feature *foot*.[5] Thus the preceding rule becomes

$$\begin{array}{l} X_0 \rightarrow X_1,\ X_2 \\ \quad \langle X_0\ head\ n \rangle\ = - \\ \quad \langle X_0\ head\ v \rangle\ = + \\ \quad \langle X_0\ head\ bar \rangle\ = 2 \\ \quad \langle X_0\ head\ subj \rangle\ = + \\ \quad \langle X_1\ head\ bar \rangle\ = 2 \\ \quad \langle X_2\ head\ subj \rangle\ = - \end{array} \qquad (R_3)$$

We use an operation add_c (read "add conservatively") which adds an equation to a PATR rule conservatively, in the sense that the equation is added only if the equations are not thereby rendered unsolvable. If addition would yield unsolvability, then a weaker *set* of unifications are added (conservatively) instead, one for each feature in the domain of the value being equated. For instance, suppose that the operation $add_c(\langle X_0\ head \rangle = \langle X_1\ head \rangle)$ is called for, where the domain of the head feature values (i.e., the various head features) are a, b, and c. If the equations in the rule already specify that $\langle X_0\ head\ a \rangle \neq \langle X_1\ head\ a \rangle$ then this operation would add only the two equations $\langle X_0\ head\ b \rangle = \langle X_1\ head\ b \rangle$ and $\langle X_0\ head\ c \rangle = \langle X_1\ head\ c \rangle$, since the addition of the given equation itself would cause rule failure. Thus the earlier constraint of values for the a feature is given precedence over the constraint to be added.

In the description of the algorithm, a nonempty path p is said to be defined for a feature structure X if and only if p is a unit path $\langle f \rangle$ and $f \in dom(X)$ or $p = \langle f\ p' \rangle$ and p' is defined for $X(f)$. Our notion of a feature's being defined for a constituent corresponds to the GPSG concepts of being instantiated *or* of covarying with some other feature.

As in the previous definition, we will be quite lax with respect to our notation for paths, using $\langle \langle a\ b \rangle\ c \rangle$ and $\langle a\ \langle b\ c \rangle \rangle$ as synonymous with $\langle a\ b\ c \rangle$. Also, we will consistently blur the distinction between a set of equations and the feature structure it determines. (See Shieber [7] for details of the mapping that makes this possible.)

3.2 The Algorithm Itself

Now our algorithm for compiling a GPSG grammar into a PATR grammar follows:

[5] But recall that *slash* is a head feature and thus would fall under the path $\langle head\ slash \rangle$.

For each ID rule of GPSG (basic or derived by metarule) $X_0 \rightarrow X_1, \ldots, X_n$:

CAP If X_i controls X_j (determined by $Type(X_i)$ and $Type(X_j)$), then $add_c(\langle X_i\ con\rangle = \langle X_j\ con\rangle)$ where

$$con = \begin{cases} \langle head\ slash\rangle & \text{if } \langle head\ slash\rangle \text{ is defined for } X_i \\ \langle head\ agr\rangle & \text{otherwise} \end{cases}$$

FFP For each foot feature path p (e.g., $\langle head\ slash\rangle$), if p is not defined for X_0 , then $add_c(\langle X_i\ p\rangle = \langle X_0\ p\rangle)$ for zero or more i such that $0 < i \leq n$ and such that p is not defined for X_i.[6]

FSD$_{lex}$ For all paths p with a default value, say, d, and for all i such that $0 < i \leq n$, if $\langle X_i\ bar\rangle = 0$ and p is not defined for X_i, then $add_c(\langle X_i\ f\rangle = d)$.

HFC For X_i the head of X_0, $add_c(\langle X_i\ head\rangle = \langle X_0\ head\rangle)$.

FSD$_{nonlex}$ For all paths p with a default value, say, d, and for all i such that $0 < i \leq n$, if $\langle X_i\ bar\rangle \neq 0$ and p is not defined for X_i, then $add_c(\langle X_i\ f\rangle = d)$.

3.3 An Example

Let us apply this algorithm to the preceding rule R_1.[7] We start with the PATR equivalent R_3. By checking the existing control relationships in this rule *as currently instantiated*, we conclude that the subject X_1 controls the head X_2. We conservatively add the unification $\langle X_2\ head\ agr\rangle = \langle X_1\rangle$. This can be safely added, and therefore is.

Next, the FFP step in the algorithm can instantiate the rule further. Suppose we choose to instantiate a slash feature on X_2. Then we add the equation $\langle X_0\ head\ slash\rangle = \langle X_2\ head\ slash\rangle$. Lexical default values require no new equations, since no constituents in the rule are given as 0 bar at this point.

The HFC conservatively adds the equation $\langle X_0\ head\rangle = \langle X_2\ head\rangle$, as X_2 is the head of X_0. But this equation, as it stands, would lead to the entire set of equations being unsolvable, since we already have conflicting values for the head feature *subj*. Thus the following set of unifications is added instead:[8]

$\langle X_0\ head\ n\rangle = \langle X_2\ head\ n\rangle$
$\langle X_0\ head\ v\rangle = \langle X_2\ head\ v\rangle$
$\langle X_0\ head\ bar\rangle = \langle X_2\ head\ bar\rangle$
$\langle X_0\ head\ agr\rangle = \langle X_2\ head\ agr\rangle$
$\langle X_0\ head\ inv\rangle = \langle X_2\ head\ inv\rangle$
...

Finally, nonlexical defaults are introduced for features not in the domains of constituents.[9] Since the path $\langle head\ inv\rangle$ is defined for the constituents X_0 and X_2,[10] the default value (i.e., '–' according to FSD 1 of Gazdar et al.) is not instantiated on either constituent. Similarly, the *case* default value (*acc*, FSD 10) is not instantiated on the subject NP. But the *conj* feature default[11] ('$\sim$') will be instantiated on all three constituents with the equations

$\langle X_0\ conj\rangle = \sim$
$\langle X_1\ conj\rangle = \sim$
$\langle X_2\ conj\rangle = \sim$

The (partial) generated rule is the following:

$X_0 \rightarrow X_1, X_2$
$\langle X_0\ head\ n\rangle = -$
$\langle X_0\ head\ v\rangle = +$
$\langle X_0\ head\ bar\rangle = 2$
$\langle X_0\ head\ subj\rangle = +$
$\langle X_1\ head\ bar\rangle = 2$
$\langle X_2\ head\ subj\rangle = -$
$\langle X_2\ head\ agr\rangle = \langle X_1\rangle$
$\langle X_0\ head\ slash\rangle = \langle X_2\ head\ slash\rangle$
$\langle X_0\ head\ n\rangle = \langle X_2\ head\ n\rangle$ (R_4)
$\langle X_0\ head\ v\rangle = \langle X_2\ head\ v\rangle$
$\langle X_0\ head\ bar\rangle = \langle X_2\ head\ bar\rangle$
$\langle X_0\ head\ agr\rangle = \langle X_2\ head\ agr\rangle$
$\langle X_0\ head\ inv\rangle = \langle X_2\ head\ inv\rangle$
...
$\langle X_0\ conj\rangle = \sim$
$\langle X_1\ conj\rangle = \sim$
$\langle X_2\ conj\rangle = \sim$
...

3.4 Problems and Extensions

Several problems have been glossed over in the previous discussion. First, we have not mentioned the role of LP rules. Two possibilities are available for their interpretation: a "run-time" and a "compile-time" interpretation. We can augment the PATR formalism with LP rules in the same way as Gazdar et al., providing for local sets of nodes to satisfy an unordered PATR rule if and only if the nodes are extensions of elements in the ID rule such that the LP rules are all satisfied. Alternatively, we can generate at compile time all possible orderings of the unordered rules compatible with the LP statements, but this leads us into the problem of interpreting LP statements relative to partially instantiated categories, an issue beyond the scope of this paper.

Second, feature cooccurrence restrictions were ignored in the previous discussion. Again, we will limit ourselves to a brief discussion of the possibilities. One alternative is to modify the lat-

[6] Several comments are pertinent to this portion of the algorithm. First, it is the FFP portion that is responsible for its nondeterminism. Second, the operation add_c is actually superfluous here. The equation can simply be added directly, since we have already guaranteed that the pertinent features are not yet instantiated. By a similar argument, we can conclude that only the add_c operations in the CAP and HFC are actually necessary. We will use add_c, however, for uniformity. Finally, we assume that an FSD will place the value $\sim$ on any remaining constituents unmarked for foot features.

[7] We do not include here the effect of the rule on every feature postulated by Gazdar et al. but only a representative sample.

[8] A more efficient representation of such sets could be achieved by the introduction of nonmonotonic operations such as overwriting or priority union. But such considerations need not concern us here.

[9] We have made the simplifying assumption that feature specification defaults are stated in terms of simple default values for features, rather than the more complex boolean conditions used in the Gazdar et al. text. The modifications to allow the more complex FSDs may or may not be straightforward.

[10] The value of the feature *head* on the constituent X_0 has the feature *inv* in its domain because the unification $\langle X_0\ head\ inv\rangle = \langle X_2\ head\ inv\rangle$ gives as value to $\langle X_0\ head\ inv\rangle$ a variable, the same variable as the value for $\langle X_2\ head\ inv\rangle$. Thus the path $\langle head\ inv\rangle$ is defined for X_0 and, similarly, for X_2.

[11] We assume here, contra Gazdar et al., that '$\sim$' is a full-fledged value in its own right, at least as interpreted in this compilation. Since this value fails to unify with any other value, e.g., '+' or '–', it has exactly the behavior desired, namely, that the feature is prohibited from taking any of its standard values.

tice of categories relative to which unification is defined[12] in such a way that all categories violating the FCRs are simply removed. Then unification over this revised lattice will be used instead of the simpler version and FCRs will automatically always be obeyed. Unfortunately, the possibility exists that unification over the revised lattice may not bear the same order-independence properties that characterize unification over the freely-generated lattice. Of course, if this turns out to be the case, it casts doubt on the well-foundedness of the original Gazdar et al. interpretation of FCRs as well, and thus is an interesting question to pursue.

Another alternative involves checking the FCRs at every point in the algorithm, throwing out any rules which violate them at any point. In addition, FCRs would be required to be checked during run-time as well. This alternative, though more direct, violates the spirit of the enterprise of giving a compilation from the complex Gazdar et al. formulation to a simpler system.

A final problem concerns the ordering of the HFC and the CAP. The definitions of controller and controllee necessary for stating the CAP depend on the assignment of semantic types to constituents, which in turn depend on the configuration of features in the categories. We have already noted that the features pertinent to the definition of semantic type (and hence control) do not include instantiated foot features. Indeed, Gazdar et al. claim that "it is just HEAD feature specifications (other than those which are also FOOT feature specifications) and inherited FOOT feature specifications that determine the semantic types relevant to the definition of control." [2, p. 87] Unfortunately, the ordering we have given precludes instantiated head features from participating in the definition of semantic type and hence the CAP.[13] It seems that the HFC must apply before the CAP for the definition of semantic type, but after the CAP so that the CAP instantiations of head features take precedence. Thus, our earlier claim of strict ordering may be falsified by this case.

Of course, the set of features necessary for type determination and the set instantiated by the CAP may be disjoint. In this case, we can merely split the application of the HFC in two, instantiating the former class before the CAP and the latter class after the FFP as originally described. Alternatively, it might be possible to notate head features on the head constituent rather than the parent as is conventially done. In this case, the information needed by the CAP is inherited, not instantiated, head feature values, and thus not subject to the ordering problem.

On the other hand, if the sets are nondisjoint, this presents a problem not only for our algorithmic analysis, but for the definition of GPSG given by Gazdar et al. Suppose that the HFC determines types in such a way that the CAP is required to apply and instantiates head features thereby overriding the original values (since the CAP takes precedence) and changing the type determination so that the CAP does not apply. We would thus require the CAP to apply if and only if it does not apply. This paradox appears as an ordering cycle in our algorithm; in the declarative definition of Gazdar et al., it would be manifested in the inadmissability of all local sets of nodes [1], an equally unattractive effect. We leave the resolution of this problem open for the time being, merely noting that it is a difficulty for GPSG in general, and not only for our characterization.

[12] For the technical background of such a move, see the discussion of PATR semantics [3].

[13] I am indebted to Roger Evans and William Keller for pointing this problem out to me and for helpful discussion of solution alternatives.

4 Conclusion

The axiomatic formulation of generalized phrase structure grammar by Gazdar et al. is a quite subtle and complex system. Yet, as we have shown, GPSG grammars can be substantially converted to grammars in a simpler, and constructive, axiomatic system through a straightforward (albeit procedural) mapping. Intrinsic in this conversion is the use of a unification-based grammar formalism, so that axioms can be stated schematically, without enumerating all of their possible instantiations. In fact, we would contend that defining the semantics of a GPSG grammar in this way yields a much simpler formulation. The need for such a reconstruction is evident to anyone who has studied the Gazdar et al. text.

Of course, even if certain parts of the GPSG formalism not discussed fully here, i.e., FCRs and LP constraints, are found not to be reducible to PATR, this in itself would be an interesting fact. It would show that exactly those portions of the formalism were truly essential for stating certain analyses, i.e., that analyses using those formal devices do so necessarily.

We find a hopeful sign in the recent work in GPSG that is proceeding in the direction of using unification directly in the rules, in addition to its implicit use in feature instantiation principles. We hope that this paper has provided evidence that such a system may be able to more simply state the kinds of generalizations that linguists claim, and has pointed out both the possibilities and difficulties inherent in these techniques.

References

[1] Gerald Gazdar. Personal communication, 1986.

[2] Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar.* Blackwell Publishing, Oxford, England, and Harvard University Press, Cambridge, Massachusetts, 1985.

[3] Fernando C. N. Pereira and Stuart M. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the Tenth International Conference on Computational Linguistics,* Stanford University, Stanford, California, 2-7 July 1984.

[4] Carl Pollard. Lecture notes on head-driven phrase-structure grammar. February 1985. Center for the Study of Language and Information, unpublished.

[5] Stuart M. Shieber. The design of a computer language for linguistic information. In *Proceedings of the Tenth International Conference on Computational Linguistics,* Stanford University, Stanford, California, 2-7 July 1984.

[6] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar. CSLI Lecture Note Series,* Center for the Study of Language and Information, Stanford, California, Forthcoming.

[7] Stuart M. Shieber. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics,* University of Chicago, Chicago, Illinois, July 1985.